MILS: Architecture for High-Assurance Embedded Computing

W. Mark Vanfleet
National Security Agency

R. William Beckwith
Objective Interface Systems

Dr. Ben Calloni
Lockheed Martin Aeronautics Company

Jahn A. Luke
Air Force Research Laboratory

Dr. Carol Taylor
University of Idaho

Gordon Uchenick
Objective Interface Systems

The Department of Defense's increasing dependence on new technology such as unmanned aerial vehicles has created a need for high-assurance systems that deliver the strongest degree of security and safety. However, there has been a notable lack of systems, commercial or government-sponsored research, and engineering that meet these requirements. The Multiple Independent Levels of Security (MILS) architecture is proposed as a solution to meet the needs for critical information assurance. MILS is a componentized architecture based on a commercial off-the-shelf separation kernel that enforces strict communication and partitioned process execution. MILS supports multiple levels of security communication, security policy composition, and modular design so that critical components are able to be evaluated at the highest levels to ensure secure and safe operation.

Information supremacy wins wars. Warfare has always required sharing the right information with the right person at the right time. Technology today enables information sharing on a scale well beyond what our forefathers imagined, but sharing information with the wrong individuals can have catastrophic consequences.

Secure information sharing is critical to enable and protect the warfighter without compromising the mission. The challenge is that warfighter-crucial information is highly diverse. Initiatives such as Network-Centric Warfare, System of Systems, and the Global Information Grid strengthen the desire to share information with multiple levels of security (MLS). MLS systems have historically been among the most challenging and expensive systems to develop and deploy [1]. Sharing and separating information in coalition force operations is an equally challenging and further complicating problem.

Multiple Independent Levels of Security (MILS) is an architecture that makes development, accreditation, and deployment of MLS-capable systems more practical, achievable, and affordable. The MILS architecture significantly increases protection, reduces time to develop, and reduces schedule risk of deploying technology to provide high-assurance systems that are both safe and secure [1].

While the MILS architecture allows for a system of highly secure distributed components, it does not automatically guarantee a secure composed system out of independent secure components. System-wide security is still up to the system designer with MILS providing the building blocks and tools needed to construct a system-level security policy that can then be verified. There are tools, e.g., Boundary Flow Modeling [2], for assuring that security policies compose for a given system.

Where We Have Been

In almost all commercial off-the-shelf (COTS) operating systems and communications technologies, security is an afterthought, addressed via a *fail-first*, *patch-later* paradigm. When a system is penetrated, fixes are then pursued to plug the hole. After a new virus propagates, the enabling weakness is repaired to stop further infection. The frail approach of attempting repair of infected systems is also common. Damage is frequently not detectable or repairable in systems with weak security foundations. Fail-first, patch-later is inappropriate for any mission-critical system because it is reactive and always one step behind the attacker. In mission-critical systems, damage must be avoided or bounded when impossible to avoid. Proactive measures are required to safeguard information and the warfighter, and prevent the damage from happening in the first place.

In traditional architectures, there were good reasons to assign all policy enforcement to a monolithic security kernel. To ensure that enforcement was non-bypassable, security functions had to be part of every system service request. To ensure that enforcement was tamper-proof, security functions had to be in an address space separate from the application [3]. These security functions needed to be in a monolithic security kernel since the computing power of two decades ago was not sufficient to perform the context switches required to separate all of these processes and data and still maintain system performance.

The security kernel with its set of trusted security functions often produced large, complex, unstructured programs that were difficult to certify at the higher assurance levels [4]. SCOMP, which managed to achieve the highest A1 security rating via the historic "Orange Book" [5][1], was based on a very simple security kernel [3]. Most security kernel-based systems never achieved the highest level of certification, which required formal verification. The Motorola Network Encryption System that handled MLS data through encryption achieved a much lower B2 rating, and the XTS-300, the successor to SCOMP, was certified at a B3 rating [3].

Aside from the difficulty of certifying systems with complex, monolithic kernels, the more important problem is in trying to enforce a single, system-wide security policy. For example, Blacker, which successfully handled encryption of MLS data, could not successfully accommodate administrative traffic within its model of classification levels [3]. In general, security policies in the kernel did not provide the robustness required for many applications where application-specific security policies would have provided more tightly focused protection.

Where We Are Going

MILS was created to enable application-level security engineering at a high level of assurance while being affordable. MILS takes advantage of Moore's Law's [6] performance increases over the last two decades by layering small, formally modeled and mathematically verified components together to create a high-assurance foundation. In MILS, applications are empowered to enforce their own security policies instead of relying on generalized kernel security services. MILS also enables efficient systems engineering, where high-assurance components can be effectively reused without modification. This lowers certification costs since certification artifacts can be reused.

The concept of MILS originated in papers written by John Rushby, Ph.D, of the Stanford Research Institute in the early 1980s [7, 8]. Rushby proposed that a separation kernel divide memory into partitions using the hardware memory management unit and allow only carefully controlled communications between non-kernel partitions. This allows one partition to provide a service to another with minimal intervention from the kernel [7].

Traditional operating system services that previously ran in privileged (i.e., supervisor) mode such as device drivers, file systems, network stacks, etc., now run in non-privileged (i.e., user) mode. Because a separation kernel provides very specific functionality, the security policies that must be enforced at this level are relatively simple. The primary concerns of a separation kernel are the partitioning of processes and data plus the containment of systemwide failures. Consequently, we can capture the security requirements for a separation kernel by four foundational security policies:

- **Data Isolation.** Information in a partition is accessible only by that partition, and private data remains private.
- **Control of Information Flow.** Information flow from one partition to another is from an authenticated source to authenticated recipients; the source of information is authenticated to the recipient, and information goes only where intended.
- **Periods Processing.** The microprocessor and any networking equipment cannot be used as a covert channel to leak information to listening third parties.
- **Fault Isolation.** Damage is limited by preventing a failure in one partition from cascading to any other partition. Failures are detected, contained, and recovered locally.

The resultant kernel is now much smaller and simpler, and conducive to rigorous inspection and mathematical proof of correctness by techniques such as formal methods. This size reduction is an instantiation of MILS's most fundamental benefit: *Dramatically reduce the amount of security-critical code so that we can dramatically increase the level of rigor when we inspect that code.* If we are doing very few things, we should be able to do them very well, so well, in fact, that the code can be *trusted* to protect our most valuable data under the highest level of threat.

MILS middleware is an expansive concept with a very broad user-mode layer. It contains many operating system services such as device drivers that previously ran in privileged mode. Running in user mode, they are subject to the kernel's security policy enforcement. MILS middleware also includes functions traditionally thought of as being one level removed from the core operating system: file systems, network stacks, common libraries, encryption, authentication, etc. MILS middleware also includes traditional application-level middleware technologies such as Common Object Request Broker Architecture (CORBA) [8], Data Distribution Service (DDS), Web services, etc. MILS middleware resides in the same user-mode partition as the application that it supports or in protected user-mode partitions by itself.

Applications do their processing and *enforce their own security policies* in user-mode partitions. Applications running in their partitions can only access the memory that has been explicitly allocated for each partition. Application partitions can only communicate with each other through paths that have been configured when the system was generated. Under no circumstances may application partitions access hardware directly unless explicitly authorized to do so. The MILS architecture, along with a notional set of allowable information flows, is illustrated in Figure 1.

Why is application-level security-policy enforcement effective in MILS when it was not effective by itself in traditional monolithic architectures? It is because the MILS separation kernel guarantees control of information flow and data isolation. It makes this guarantee for the first time at an assurance level that was next to impossible to achieve with the monolithic kernels. Due to technology advances in both smaller circuits and increased functionality, we now have processors powerful enough to handle the context switching required for MILS, while still maintaining system performance.

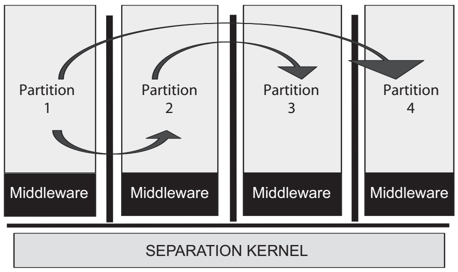
In the last 15 years, the number of context switches per unit of time that a state-of-the-art microprocessor can handle has increased by a factor of 1,000. Processor speed has increased by a factor of 100. The number of transistors per cubic inch on a wafer has increased by a factor of 125. We can now perform 50,000 context switches at a cost of 5 percent of the microprocessor clock. This 5 percent is the MILS security and safety tax.

Because information originates only from authorized sources, is delivered only to the intended recipients, and the source is authenticated to that recipient, the application developer is empowered to build his or her own reference monitors[2] at the application layer that include the following:

- **Non-bypassable.** Security functions cannot be circumvented.
- **Evaluatable.** Security functions are small and simple enough to enable rigorous proof of correctness through mathematical verification.
- **Always Invoked.** Security functions are invoked each and every time.
- **Tamperproof.** Security functions and their data cannot be modified without authorization, either by subversive or poorly written code.

An acronym for these four attributes is *NEAT* [9]. Security policy enforcement that is not NEAT is not effective. Although other operating systems have offered some form of non-bypassability and tamperproof functionality, MILS provides

Figure 1: *MILS Architecture Information Flows*



Common Terms

API	Application Programming Interface
CIK	Crypto Ignition Key
CORBA	Common Object Request Broker Architecture
DDS	Data Distribution Service
EAL	Evaluation Assurance Level
HAL	Hardware Abstraction Layer
HTTP	Hyper-Text Transfer Protocol
IPv6	Internet Protocol version 6
MILS	Multiple Independent Levels of Security
MLS	Multiple Levels of Security
MMU	Memory Management Unit
MSLS	Multiple Single Levels of Security
PCS	Partitioning Communications System
SOAP	Simple Object Access Protocol
RTOS	Real-Time Operating System
TCP/IP	Transport Control Protocol/ Internet Protocol
UDDI	Universal Description, Discovery, and Integration
WSDL	Web Services Description Language
XML	Extensible Markup Language

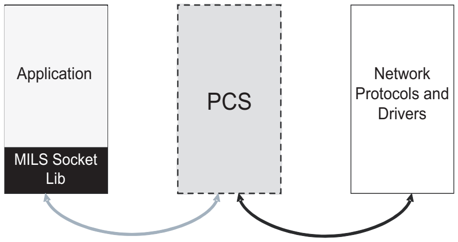


Figure 2: *Secure Network System Configuration*

*NEAT*ness for the first time in a COTS package that is formally modeled and mathematically verified at a high assurance level.

Divide and Conquer

The duration, schedule risk, and cost of evaluating, certifying, and deploying software increase non-linearly with the size of the code. These increases are especially onerous at high levels of assurance. Guaranteed NEATness enables us to design a MLS or Multiple Single Levels of Security (MSLS)[3] system as a set of independent system high partitions with cross-domain servers, downgraders, and guards enabling secure communications both among those partitions and also with external systems.

Another MILS objective is to enable the evaluation and certification of a complex system to be broken down into a number of independent, small evaluations. Security-critical software components that handle more than one level of information can be evaluated at high levels of robustness, approximately Evaluation Assurance Level (EAL) 6+ of the Common Criteria, an internationally approved set of security standards [10]. Cross-domain servers, downgraders, and guards, leveraging NEATness, can be small and tightly focused, making high-assurance evaluations of those components practical, achievable, and affordable. Single-level partitions, which each deal with only one level of information, can be evaluated at medium levels of robustness, approximately EAL 4, which is practical and achievable for large bodies of code. The independence of these evaluations also enables reuse of code, reuse of application programming interfaces (APIs), reuse of specifications, reuse of evaluation artifacts, and reuse of certifications to the greatest degree possible.

Connecting to Other Systems

MILS network components such as protocol stacks and their associated interface device drivers can be put into partitions of their own. This architecture has several advantages:

- Network facilities can be used by multiple application partitions.
- Network data is processed in unprivileged user mode, eliminating a vulnerability that is a common avenue of attack.
- Complex protocol code such as Internet Protocol (IP) Ver. 6 can be evaluated and certified independent of the applications using the code, enabling reuse of the evaluation artifacts.

Applications use an API to interact with the network. The MILS network API can have the same semantics as in a traditional operating system such as the familiar Transport Control Protocol/IP socket calls. The API implementation difference can be completely *under the hood*, transparent to the application developer. Instead of interacting directly with the protocol, a MILS socket implementation uses the separation kernel's interpartition communications facility to forward outgoing data to the protocol stack. Incoming data is handled similarly in the opposite direction.

Secure Network Systems

The network is the platform. The embedded computer that is not connected to another processor is a rare exception. By enforcing its four foundational security policies, MILS implements a robust information assurance foundation in a single node. We can then implement a robust information assurance foundation throughout a distributed system by providing *end-to-end* enforcement of those same security policies. End-to-end enforcement is provided by a high-assurance middleware component called the Partitioning Communications System (PCS). Leveraging the separation kernel's guarantee of controlled information flow within a single node, the PCS is always interposed between an application and the protocols/drivers that effect an off-board data transfer. The configuration is illustrated in Figure 2.

The PCS enforces the security policies end-to-end by providing the following:

- Strong identity of each node within a collection of MILS nodes (an enclave).
- Separation by level and/or community of interest. Enclaves are then connected together via high-assurance MILS cross-domain servers.
- Secure configuration, validating that all security databases are consistent.
- Secure image loading.
- Secure clock synchronization.
- Provisioning of bandwidth and quality of service.
- Suppression of covert channels.

Network Middleware Tools

While we are developing new systems that enable the warfighter to share information, it is important to not reinvent the wheel. Distributed system solution designers make frequent use of COTS network middleware for various application paradigms:

- Client/server, often using CORBA [11], distributes logic.
- Publish/subscribe, often using DDS, distributes data.
- Web-enabled services, using HyperText Transfer Protocol servers and components such as Extensible Markup Language; SOAP [Simple Object Access Protocol]; Web Services Description Language; Disco; Universal Description, Discovery and Integration; etc. that enable the communication between large diverse distributed communities of interest.

All of these technologies can be viewed as tools that provide the application programmer with a higher level abstraction to the rudimentary socket interface. Much of the code for these networking middleware technologies can either reside together in the same partition as the application that it supports, or it can reside in a partition by itself. In either case, porting existing code to a MILS environment is a straightforward task because the socket API does not need to change. The PCS still fits between the network middleware and the protocol stack and/or device drivers.

The system architect should not view the use of CORBA, DDS, or Web services as mutually exclusive. A single application can use CORBA for remote invocation, for distribution of logic, and for *smart pull* of needed information; it can use DDS for *smart push* of sensor data that is being monitored; and it can use Web services as a graphical interface for reports from one large community of interest to another, e.g., the Army infantry reporting threat data and the Air Force monitoring Web reports and providing air support.

There is an interesting side benefit to combining network middleware with the PCS. One of the purposes of network middleware is to make the number and location of processors sharing traffic as transparent as possible to the application. At the same time, in a secure networked system, we need to know exactly where our data came from and where it will go. Merging PCS functions with network middleware such as CORBA or DDS gives the system designer the flexibility to relocate system functions without introducing new threats to data confidentiality or integrity. This enables ad hoc networks and coalitions to be formed based on newly identified threat data, and to be dissolved as soon as the threat is dealt with.

Security is required when fielding systems that are either mission-critical or use national information. At the same time, there is a massive investment in applications using traditional operating systems and traditional middleware. The MILS architecture can provide a high-assurance foundation for fielded systems while preserving much of the legacy code base.

Guest Operating Systems

A traditional operating system, either an embedded real-time operating system (e.g., INTEGRITY, VxWorks, or LynxOS) or a desktop operating system (e.g., Linux, Windows, or Solaris) can run inside a MILS partition as a *guest operating system*. Operating systems written with portability in mind have a hardware abstraction layer (HAL) that localizes all processor-specific functions. Writing a new HAL is the major task in porting to a new central processing unit. You can use that same expertise to write a HAL that abstracts the MILS separation kernel as the hardware to the guest.

By itself, the guest operating system concept enables legacy applications to be easily ported to the high-assurance MILS environment. Another possibility is that multiple MILS partitions can each contain an instance of the guest operating system. This effectively creates multiple virtual operating systems on a single real microprocessor. The MILS separation kernel provides *trustworthy* separation with respect to both memory access and central processing unit time. Communications among the partitions is limited to those paths explicitly created when the system was generated. This is a practical path to implementation of cross domain solutions. It is also a practical path to implementation of high-assurance workstations suitable for MLS or coalition force operations.

For example, inside a partition the guest operating system can run as a thin client; it can be downloaded from a remote server. Which remote server a thin client is downloaded from can be determined from a token reader or crypto ignition key. The token would indicate nationality, clearance, and job title. The PCS would open a secure connection to a server that the user, who inserted and unlocked the token, was authorized to communicate with. Access to the local devices such as screen, keyboard, mouse, hard drive, etc., would be provided by the MILS workstation.

Supporting the Warfighter

The Air Force Research Laboratory (Information Directorate), in cooperation with the National Security Agency, Department of Defense prime contractors, academia, and software suppliers, is managing a MILS program to combine the best of existing commercial standards for flight safety and integrated modular avionics with the following:

- DO-178B, Software Considerations in Airborne Systems and Equipment Certification, Level A [12].
- ARINC-653, Avionics Application Software Standard Interface [13].

The program is also combined with the following appropriate standards for security:

- Common Criteria (International Organization for Standardization 15408), EAL 6, augmented [10].
- Director of Central Intelligence Directive 6/3, Protecting Sensitive Compartmented Information Within Information Systems, Protection Level 5 [14].

There is significant synergy among these standards. While they each have a specific area of interest, there is a great deal of common ground between safety-critical and security standards with respect to sound engineering practice, meeting requirements, and having the plans in place to address flaws.

The participating software suppliers that are currently developing MILS separation kernels are, alphabetically, Green Hills Software, Inc. [15], LynuxWorks, Inc. [16], and Wind River Systems, Inc. [17]. Objective Interface Systems, Inc. [18] is currently developing the partitioning communications system.

Putting It All Together

MILS is all about keeping things separate that need to be separate and doing so with components that we can trust with our most important data under the most severe threat. For security, we are keeping data separate by classification level, by community of interest, and by nationality. For safety, we are keeping applications separate by level of criticality. All of this is done with COTS software and certification artifacts that are reusable. Leveraging this reusability makes MSLS/MLS system development practical and certification/accreditation affordable and achievable. The end result is fielded systems that have high-assurance foundations but do not require custom-built security architectures for each new system.

For more information about MILS, please see <http://mils.ois.com>.◆

References

1. Vanfleet, Willard Mark, et al. "An Architecture for Deeply Embedded, Provable High Assurance Applications." May 2003.
2. Freeman, James, George Dinolt, and Richard Neely. An Internet System Security Policy and Formal Model. Proc. of 11th National Computer Security Conference, Oct. 1988: 10-19.
3. Anderson, Ross. Security Engineering. New York: Wiley & Sons, 2001.
4. Rushby, John. A Trusted Computing Base for Embedded Systems. Proc. of the 7th Department of Defense/NBS Computer Security Conference, Sept. 1984: 294-311.
5. Department of Defense. Trusted Computer System Evaluation Criteria (The Orange Book). DoD 5200.28-STD. Washington: DoD, 1983.
6. Moore, G. "Cramming More Components Onto Integrated Circuits." Electronics Magazine 19 Apr. 1965.
7. Rushby, John. "The Design and Verification of Secure Systems." ACM Operating Systems Review 15.5.
8. Rushby, John. "Proof of Separability: A Verification Technique for a Class of Security Kernels." Computer Science 137: (1982) 352-367.
9. Partitioning Kernel Protection Profile, May 2003.
10. Common Criteria for Information Technology Security Evaluation, Ver. 2.1, 19 Sept. 2000.
11. Currey, Jonathan, Bill Beckwith, et al. Real-Time CORBA 1.1 Specification, Object Management Group Aug. 2002.
12. RTCA <www.rtca.org>.
13. ARINC <www.arinc.com/cf/store/index.cfm>.
14. Director of Central Intelligence. "Protecting Sensitive Compartmented Information Within Information Systems." Directive 6/3. Washington: DCID, 5 June 1999 <www.fas.org/irp/offdocs/DCID_6-3_20Policy.htm>.
15. Green Hills Software <www.ghs.com>.
16. Lynux Works <www.lynuxworks.com>.
17. Wind River Systems <www.windriver.com>.
18. Objective Interface <www.ois.com>.

Notes

[1] Orange book levels began with A1 and moved down through levels B3, B2, B1, C2, and C1.

[2] A reference monitor is an Access Control concept referring to an abstract machine that mediates all accesses to objects by subjects [3].

[3] MSLS means there are multiple channels each with their own separate data classification [9].

About the Authors

W. Mark Vanfleet has worked for the National Security Agency (NSA) Information Assurance Directorate for 18 years as an information systems security analyst and mathematician. He holds NSA certifications in crypto-mathematics, communication and information systems security, and software engineering process and practice. Vanfleet has been involved in high-assurance software architecture, design, and evaluation for 25 years. He has bachelor's degrees in mathematics and computer science, and a master's degree in mathematics and statistics from the University of Utah.

National Security Agency
9800 Savage RD STE 6709
Fort Meade, MD 20755-6709
Phone: (410) 854-6361
E-mail: wvanflee@restarea.ncsc.mil

Jahn A. Luke is a senior program manager at the Embedded Information Systems Branch, Information Directorate, Air Force Research Laboratory (AFRL) where he is lead for legacy system modernization and manager of the Multiple Independent Levels of Security (MILS) program. He has more than 29 years of hardware and software experience at AFRL in the development of technologies for real-time simulation systems and embedded software systems. In addition to MILS, he currently manages projects addressing the upgrade of legacy embedded computer systems for such programs as F/A-22, CV-22, and F-117. Luke has a Bachelor of Electrical Engineering from the University of Detroit.

AFRL/IFTA
2241 Avionics CIR BLDG 620
Wright Patterson, OH 45433
Phone: (937) 255-6653 ext. 3585
Fax: (937) 656-4277
E-mail: jahn.luke@wpafb.af.mil

R. William Beckwith is the chief executive officer and chief technology officer of Objective Interface Systems, Inc. He has been engineering software for over two decades with the last decade focused on embedded and real-time systems. Beckwith is a frequent speaker on real-time, embedded, and high-assurance software in North America, Japan, and Europe. He is currently involved in developing software and standards for high-assurance embedded systems. Beckwith continues to lead the initiative for the advancement of Common Object Request Broker Architecture and Multiple Independent Levels of Security in the real-time and embedded communities.

Objective Interface Systems
13873 Park Center RD STE 360
Herndon, VA 20171-3247
Phone: (703) 295-6519
Fax: (703) 295-6501
E-mail: bill.beckwith@ois.com

Carol Taylor, Ph.D. is a professor of computer science at the University of Idaho where she currently teaches classes in computer security and does research. Taylor's research interests are in computer security and software engineering with a special emphasis on high-assurance systems. Her research background includes projects in survivability, intrusion detection, formal methods, and multiple levels of security policy. Prior to obtaining her doctorate, Taylor held programming/analyst positions.

University of Idaho
Computer Science Department
Moscow, ID 83844
Phone: (208) 885-5276
Fax: (208) 885-9052
E-mail: ctaylor@cs.uidaho.edu

Ben Calloni, Ph.D., is a senior software and avionics researcher for Lockheed Martin (LM) Aeronautics Company in Fort Worth, Texas. He has been addressing the feasibility of using standards-based commercial software in mission-critical avionics systems since joining LM in 1997. The past three years he has been investigating the use of commercial off-the-shelf-based security solutions (Multiple Independent Levels of Security program) for the various LM Aero weapon systems. Calloni serves on the board of directors for and is active in both the Object Management Group and The Open Group: International Standards Consortia. He has degrees in industrial engineering and computer science from Purdue University and a doctorate in computer science from Texas Tech University.

Lockheed Martin Aeronautics Co.
P.O. Box 748
MZ 8604
Fort Worth, TX 76101
Phone: (817) 935-4482
Fax: (817) 762-6784
E-mail: ben.a.calloni@lmco.com

Gordon Uchenick is a frequent presenter and lecturer on Multiple Independent Levels of Security (MILS) as well as an author on the subject. Uchenick also participates in the MILS community standards bodies such as The Open Group. Prior to joining Objective Interface, he was an engineering specialist with Wind River Systems, concentrating on the company's security technologies.

Objective Interface Systems
13873 Park Center RD STE 360
Hendon, VA 20171-3247
Phone: (410) 256-7102
Cell: (410) 952-2739
Fax: (410) 256-7104
E-mail: gordon.uchenick@ois.com